1       TUNABLE NARROW-BAND FILTER INCLUDING

2              SIGMA-DELTA MODULATOR

3

4              Christopher H. Dick

5              Frederic J. Harris

6

7   CROSS-REFERENCE TO RELATED APPLICATION

8       This is a continuation-in-part of U.S. Patent Application

9   Serial No. 09/394,123 filed 9/10/1999 entitled "Narrow Band

10  Filter Including Sigma-Delta Modulator Implemented in a

11  Programmable Logic Device", incorporated herein by reference.

12

13  INTRODUCTION

14      While $\Sigma\Delta M$ techniques are applied widely in analog

15  conversion sub-systems, both analog-to-digital (ADC) and

16  digital-to-analog (DAC) converters, these methods have

17  enjoyed much less exposure in the broader application

18  domain, where flexible and configurable solutions,

19  traditionally supplied via a software DSP (soft-DSP), are

20  required. And this limited level of exposure is easy to

21  understand. Most, if not all, of the efficiencies and

22  optimizations afforded by $\Sigma\Delta M$ are hardware oriented and so

23  cannot be capitalized on in the fixed precision pre-defined

24  datapath found in a soft-DSP processor. This limitation, of

25  course, does not exist in a field programmable gate array

26  (FPGA) DSP solution. With FPGAs the designer has complete

27  control of the silicon to implement any desired datapath and

28  employ optimal word precisions in the system with the

29  objective of producing a design that satisfies the

30  specifications in the most economically sensitive manner.

1     While implementation of a digital $\Sigma\Delta$ ASIC (application-

2 specific integrated circuit) is of course possible, economic

3 constraints make the implementation of such a building block

4 that would provide the flexibility, and be generic enough to

5 cover a broad market cross-section, impractical. FPGA-based

6 hardware provides a solution to this problem. FPGAs are off-

7 the-shelf commodity items that provide a silicon feature set

8 ideal for constructing high-performance DSP systems. These

9 devices maintain the flexibility of software-based

10 solutions, while providing levels of performance that match,

11 and often exceed ASIC solutions.

12     There is a rich and expanding body of literature

13 devoted to the efficient and effective implementation of

14 digital signal processors using FPGA based hardware. More

15 often than not, the most successful of these techniques

16 involves a paradigm shift away from the methods that provide

17 good solutions in software programmable DSP systems.

18     This paper reports on the rich set of design

19 opportunities that are available to the signal processing

20 system designer through innovative combinations of $\Sigma\Delta$M

21 techniques and FPGA signal processing hardware. The

22 applications considered include narrow-band filters, both

23 single-rate and multi-rate, DC canceler, and $\Sigma\Delta$M hybrid

24 digital-analog control loops for simplifying carrier

25 recovery, timing recovery, and AGC (automatic gain control)

26 loops in a digital communication receiver.

27     This application is organized as follows: section 2

28 presents a brief overview of FPGA architecture. In section 3

29 a simple single-loop base-band $\Sigma\Delta$ modulator is introduced.

30 The structure is an extended to a normal architecture that

1  permits center frequency tuning, as well as a method for

2  working with the system degrees of freedom to trade-off

3  modulator bandwidth with dynamic range. The tunable $\Sigma\Delta M$ is

4  then utilized for implementing area efficient FPGA FIR

5  filters. The process for computing the modulator

6  coefficients for low pass, bandpass, and high pass designs

7  is described. In section 4, a new $\Sigma\Delta M$ architecture is

8  described that provides a very simple method for tuning

9  using only a single coefficient. In any fixed-point data

10 path, careful consideration must be given to the DC aspects

11 of the design. For example, the introduction of a DC

12 complement to two-step altercation between the stages of a

13 multi-stage, multi-rate filter can be problematic, causing

14 arithmetic saturation or increasing the bit error rates in

15 additional receiver. Section 5 describes a unique $\Sigma\Delta M$

16 approach to building a DC canceler. In section 6, $\Sigma\Delta M$

17 methods are described for simplifying the implementation of

18 hybrid digital-analog control loops in a system such as a

19 software-defined radio. In section 7 some comments on the

20 industrial implications of the techniques considered in the

21 application are presented. Finally, some conclusions are

22 drawn in section 8.

23      Semiconductor vendors, such as Xilinx, Altera, Atmel,

24 and AT&T, provide a range of FPGAs. The architectural

25 approaches are as diverse as there are manufacturers, but

26 some generalizations can be made. Most of the devices are

27 basically organized as an array of logic elements and

28 programmable routing resources used to provide the

29 connectivity between the logic elements, FPGA I/O pins and

30 other resources, such as on-chip memory. The structure and

1    complexity of the logic elements, as well as the

2    organization and functionality supported by the

3    interconnection hierarchy, distinguish the devices. Other

4    device features, such as block memory and delay locked loop

5    technology, are also significant factors that influence the

6    complexity and performance of an algorithm that is

7    implemented using FPGAs.

8        A logic element usually consists of one or more RAM

9    (random access memory) n-input look-up tables, where n is

10   between three and 6, in one to several flip-flops. There may

11   also be additional hardware support in each element to

12   enable high-speed arithmetic operations. This generic FPGA

13   architecture is shown in Figure 1. Also illustrated in the

14   Figure (as wide lines) are several connections between logic

15   elements and the device input/output (I/O) ports.

16   Application-specific circuitry is supported in the device by

17   downloading a bit stream into SRAM (static random access

18   memory) based configuration memory. This personalization

19   database defines the functionality of the logic elements, as

20   well as the internal routing. Different applications are

21   supported on the same FPGA hardware platform by configuring

22   the FPGA(s) with appropriate bit streams. As a specific

23   example, consider the Xilinx Virtex™ series of FPGAs. The

24   logic elements, called slices, essentially consistent to

25   four-input look-up tables (LUTs), to flip-flops, several

26   multiplexors and some additional silicon support that allows

27   the efficient implementation of carry-chains for building

28   high-speed matters, subtracters, and shift registers. Two

29   slices form a configurable logic block (CLB) as shown in

30   Figure 2. The CLB is the basic tile is used to build the

31   logic matrix. Some FPGAs, like the Xilinx Virtex families,

1  supplying on-chip block RAM. Figure 3 shows the CLB matrix

2  that defines a Virtex FPGA. Current generation Virtex

3  silicon provides a family of devices offering 768 to 12,288

4  logic slices, and from 8 to 32 variable form factor block

5  memories.

6      Xilinx XC4000 and Virtex devices also all-out the

7  designer to use the logic element LUTs as memory-either ROM

8  or RAM. Constructing memory with this distributed memory

9  approach can yield access bandwidths in many pans at GB per

10 second range.

11     Typical clock frequencies for current generation

12 devices are in the multiple tenants of megahertz (100 to

13 200) range.

14     In contrast to the logic slice architecture employed in

15 Xilinx Virtex devices, a logic block architecture employed

16 in the Atmel AT40K FPGA is shown in Figure 4. Like the

17 Xilinx device, combinational logic is realized using look-up

18 tables. In this case, to three-input LUTs and a single flip-

19 flop are available in each logic cell. The pass gates in a

20 cell form part of the signal routing network and are used

21 for connecting signals to the multiple horizontal and

22 vertical bus plains. In addition to the orthogonal routing

23 resources, indicated as N, S, E and W in Figure 4, a

24 diagonal group of interconnects (NW, NE, SE, and SW),

25 associated with each cell x output, are available to provide

26 efficient connections to neighboring cell's x bus inputs.

27     The objective of the FPGA/DSP architect is to formulate

28 algorithmic solutions for applications that best utilize

29 FPGA resources to achieve the required functionality. This

30 is a three-dimensional optimization problem in power,

31 complexity, and bandwidth. The remainder of this application

1   describes some novel FPGA solutions to several signal

2   processing problems. The results are important in industrial

3   context because they enable either smaller, and hence more

4   economic, solutions to important problems, or allow more

5   arithmetic compute power to be realized with a given area of

6   silicon.

7

8

9   $\Sigma\Delta$ MODULATORS, FIR FILTERS AND FPGAS

10      $\Sigma\Delta$M-based DSP is employed to generate FPGA hardware

11  implementations of narrow-band filters, a DC canceller, and

12  hybrid digital-analog control loops for a software-defined

13  radio architecture.

14

15      This section describes a method employing sigma-delta

16  modulation ($\Sigma\Delta$M) techniques for implementing area efficient

17  finite impulse response (FIR) filters using FPGA hardware.

18  Before treating the FPGA filter design, a brief review of $\Sigma\Delta$

19  modulation encoding is presented.

20

21  $\Sigma\Delta$ Modulation

22      Sigma-Delta modulation is a source coding technique

23  most prominently employed in analog-to-digital and digital-

24  to-analog converters. In this context, hybrid analog and

25  digital circuits are used in the realization. Figure 5 shows

26  a single-loop $\Sigma\Delta$ modulator. Provided the input signal is

27  busy enough, the linearized discrete time model of Figure 6

28  can be used to illustrate the principle. In Figure 6, the 1-

29  bit quantizer is modeled by an additive white noise source

1   with variance $\sigma_e{}^2=\Delta^2/12$, where $\Delta$ represents the quantization

2   interval. The $z$-transform

3   of the system is

4

5   Equations 1 and 2:

$$
\begin{aligned}
Y(z) &= \frac{H(z)}{1+H(z)}X(z) + \frac{1}{1+H(z)}Q(z) \\
&= H_s(z)X(z) + H_n(z)Q(z)
\end{aligned}
$$

6   where

7

8   Equation 3:

$$
H(z) = \frac{1}{z-1}
$$

9

10   which is the transfer function of delay and an ideal

11   integrator, and $H_s(z)$ and $H_n(z)$ are the signal and noise

12   transfer functions (NTF) respectively. In a good $\Sigma\Delta$

13   modulator, $H_n(\omega)$ will have a flat frequency response in the

14   interval $|f| \leq B$. In contrast, $H_n(\omega)$ will have a high

15   attenuation in the frequency band $|f| \leq B$ and a "don't care"

16   region in the interval $B < |f| < f_s/2$. For the single loop $\Sigma\Delta$

17   in Figure 6, $H_s(z) = z^{-1}$ and $H_n(z) = 1-z^{-1}$. Thus the input

18   signal is not distorted in any way by the network and simply

19   experiences a pure delay from input to output. The

20   performance of the system is determined by the noise

21   transfer function $H_n(z)$, which is given by

22

23   Equation 4:

$$
|H_n(f)| = 4\left|\sin\frac{\pi f}{f_s}\right|
$$

1

2  and is shown in Figure 7. The in-band quantization noise

3  variance is

4

5  Equation 5:

$$\sigma_n^2 = \int\limits_{-B}^{+B} | H_n(f) |^2 \, S_q(f) df$$

6

7  where $S_q(f) = \sigma_q^2 / f_s$ is the power spectral density of the

8  quantization noise. Observe that for a non-shaped noise (or

9  white) spectrum, increasing the sampling rate by a factor of

10  2, while keeping the bandwidth B fixed, reduces the

11  quantization noise by 3 dB. For a first order $\Sigma\Delta$M it can be

12  shown that

13

14  Equation 6:

$$\sigma_n^2 \approx \frac{1}{3} \pi^2 \sigma_q^2 \left( \frac{2B}{f_s} \right)^3$$

15

16  for $f_s \gg 2B$. Under these conditions doubling the sampling

17  frequency reduces the noise power by 9 dB, of which 3 dB is

18  due to the reduction in $S_q(f)$ and a further 6 dB is due to

19  the filter characteristic $H_n(f)$. The noise power is reduced

20  by increasing the sampling rate to spread the quantization

21  noise over a large bandwidth and then by shaping the power

22  spectrum using an appropriate filter.

23

24  Reduced Complexity Filters Using $\Sigma\Delta$ Modulation Techniques

1    $\Sigma\Delta M$ techniques can be employed for realizing area

2    efficient narrowband filters in FPGAs. These filters are

3    utilized in many applications. For example, narrow-band

4    communication receivers, multi-channel RF surveillance

5    systems and for solving some spectrum management problems.

6        A uniform quantizer operating at the Nyquist rate is

7    the standard solution to the problem of representing data

8    within a specified dynamic range. Each additional bit of

9    resolution in the quantizer provides an increase in dynamic

10   range of approximately 6dB. A signal with 60dB of dynamic

11   range requires 10 bits, while 16 bits can represent data

12   with a dynamic range of 96dB.

13       While the required dynamic range of a system fixes the

14   number of bits required to represent the data, it also

15   affects the expense of subsequent arithmetic operations, in

16   particular multiplications. In any hardware implementation,

17   and of course this includes FPGA based DSP processors, there

18   are strong economic imperatives to minimize the number and

19   complexity of the arithmetic components employed in the

20   datapath. An embodiment of the invention employs noise-

21   shaping techniques to reduce the precision of the input data

22   samples to minimize the complexity of the multiply-

23   accumulate (MAC) units in the filter. The net result is a

24   reduction in the amount of FPGA logic resources required to

25   realize the specified filter.

26       Consider the structure shown in Figure 8. Instead of

27   applying the quantized data x(n) from the analog-to-digital

28   converter directly to the filter, data x(n) is pre-processed

29   by a $\Sigma\Delta$ modulator. The re-quantized input samples $\hat{x}(n)$ are

30   represented using fewer bits per sample, so permitting the

1  subsequent filter H(z) to employ reduced precision

2  multipliers in the mechanization. The filter coefficients

3  are still kept to a high precision.

4      The $\Sigma\Delta$ data re-quantizer is based on a single loop

5  error feedback sigma-delta modulator shown in Figure 9. In

6  this configuration, the difference between the quantizer

7  input and output sample is a measure of the quantization

8  error, which is fed back and combined with the next input

9  sample. The error-feedback sigma-delta modulator operates on

10 a highly oversampled input and uses the unit delay $z^{-1}$ as a

11 predictor. With this basic error-feedback modulator, only a

12 small fraction of the bandwidth can be occupied by the

13 required signal. In addition, the circuit only operates at

14 baseband. A larger fraction of the Nyquist bandwidth can be

15 made available and the modulator can be tuned if a more

16 sophisticated error predictor is employed. This requires

17 replacing the unit delay with a prediction filter P(z). This

18 generalized modulator is shown in Figure 10.

19     The operation of the re-quantizer can be understood by

20 considering the transform domain description of the circuit.

21 This is expressed as

22

23 Equation 7:

$$\hat{X}(z) = X(z) + Q(z)(1 - P(z)z^{-1})$$

24

25 where Q(z) is the z-transform of the equivalent noise source

26 added by the quantizer q(·), P(z) is the transfer function of

27 the error predictor filter, and X(z) and $\hat{X}(z)$ are the

28 transforms of the system input and output respectively. P(z)

29 is designed to have unity gain and leading phase shift in

1    the bandwidth of interest. Within the design bandwidth, the

2    term $Q(z)(1-P(z)z^{-1})=0$ and so $X(z) = \hat{X}(z)$. By designing $P(z)$

3    to be commensurate with the system passband specifications,

4    the in-band spectrum of the re-quantizer output will ideally

5    be the same as the corresponding spectral region of the

6    input signal.

7         To illustrate the operation of the system consider the

8    task of recovering a signal that occupies 10% of the

9    available bandwidth and is centered at a normalized

10   frequency of 0.3Hz. The stopband requirement is to provide

11   60 dB of attenuation. Figure 11A shows the input test

12   signal. It comprises an in-band component and two out-of-

13   band tones that are to be rejected. Figure 11B is a

14   frequency domain plot of the signal after it has been re-

15   quantized to 4 bits of precision by a $\Sigma\Delta$ modulator employing

16   an 8th order predictor in the feedback path. Notice that the

17   60dB dynamic range requirement is supported in the bandwidth

18   of interest, but that the out-of-band SNR has been

19   compromised. This is of course acceptable, since the

20   subsequent filtering operation will provide the necessary

21   rejection. A 160-tap filter $H(z)$ satisfies the problem

22   specifications. The frequency response of $H(z)$ using 12-bit

23   filter coefficients is shown in Figure 11C. Finally, $H(z)$ is

24   applied to the reduced sample precision data stream $\hat{X}(z)$ to

25   produce the spectrum shown in Figure 11D. Observe that the

26   desired tone has been recovered, the two out-of-band

27   components have been rejected, and that the in-band dynamic

28   range meets the 60 dB requirement.

29

30   Prediction Filter Design

11

1    '   The design of the error predictor filter is a signal

2    estimation problem. The optimum predictor is designed from a

3    statistical viewpoint. The optimization criterion is based

4    on the minimization of the mean-squared error. As a

5    consequence, only the second-order statistics

6    (autocorrelation function) of a stationary process are

7    required in the determination of the filter. The error

8    predictor filter is designed to predict samples of a band-

9    limited

10   white noise process $N_{xx}(\omega)$   shown in Figure 12.

11   $N_{xx}(\omega)$ is defined as:

12

13   Equation 8:

14                          $$N_{XX}(\omega) = \begin{cases} 1 & -\theta \leq \omega \leq \theta \\ 0 & \text{otherwise} \end{cases}$$

15

16   and related to the autocorrelation sequence $r_{xx}(m)$   by

17   discrete-time Fourier transform (DTFT).

18

19   Equation 9:

20
                          $$N_{xx}(\omega) = \sum_{n=-\infty}^{\infty} r_{xx}(k) e^{-jwn}$$
21

22

23   The autocorrelation function $r_{xx}(n)$ is found by taking the

24   inverse DTFT of the equation immediately above.

25

26   Equation 10:

27                          $$r_{xx}(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} N_{xx}(\omega) e^{-j\omega n} d\omega$$

28

29

1   $N_{xx}(\omega)$ is non-zero only in the interval $-\theta \le \omega \le \theta$ giving $r_{xx}(n)$

2   as:

3   Equation 11:

$$r_{xx}(n) = \frac{\theta}{\pi} \operatorname{sinc}(\theta n)$$

4

5

6       So the autocorrelation function corresponding to a band-

7   limited white noise power spectrum is a sinc function. Samples

8   of this function are used to construct an autocorrelation

9   matrix which is used in the solution of the normal equations

10  to find the required coefficients. Leaving out the scaling

11  factor in the immediately above equation, the required

12  autocorrelation function $r_{xx}(n)$, truncated to p samples, is

13  defined as:

14

15   Equation 12:

$$r_{xx} = \frac{\sin(n\theta)}{n\theta} \qquad n = 0,...,p-1$$

16

17

18      The normal equations are defined as:

19  Equation 13:

20

$$r_{xx}(m) = \sum_{k=1}^{p} a(k)\, r_{xx}(m-k) \qquad m = 1,2,....,p$$   21

22

23      This system of equations can be compactly written in

24  matrix form by first defining several matrices.

25      To design a p-tap error predictor filter first compute a

26  sinc function consisting of p+1 samples and construct the

27  autocorrelation matrix $R_{xx}$ as:

28  Equation 14:

$$R_{xx} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & ... & r_{xx}(p-1) \\ r_{xx}(1) & r_{xx}(0) & ... & r_{xx}(p-2) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ r_{xx}(p-1) & r_{xx}(p-2) & ... & r_{xx}(0) \end{bmatrix}$$

Next, define a filter coefficient row-vector A as:

Equation 15;

$$A = [a(0), a(1), ..., a(p-1)]$$

where $a(i), i = 0, ...,$ p-1, are the predictor filter coefficients. Let the row-vector $R'_{xx}$ be defined as:

Equation 16:

$$R'_{xx} = [r_{xx}(1), r_{xx}(2), ..., r_{xx}(p)]$$

The matrix equivalent of equation 13 is:

Equation 17: $\qquad R_{xx} A^T = (R'_{xx})^T$

The filter coefficients are therefore given as:

Equation 18:

$$A^T = R_{xx}^{-1} (R'_{xx})^T$$

For the case in-hand, the solution of equation 18 is an ill-conditioned problem. To arrive at a solution for A, a small constant $\varepsilon$ is added to the elements along the diagonal

1    of the autocorrelation matrix $R_{xx}$ in order to raise its

2    condition number. The actual autocorrelation matrix used to

3    solve for the predictor filter coefficients is:

4

5    Equation 19:

6

7

8

9

10

11

12

$$R_{xx} = \begin{bmatrix} r_{xx}(0)+\varepsilon & r_{xx}(1) & \ldots & r_{xx}(p-1) \\ r_{xx}(1) & r_{xx}(0)+\varepsilon & \ldots & r_{xx}(M-2) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ r_{xx}(p-1) & r_{xx}(p-2) & \ldots & r_{xx}(0)+\varepsilon \end{bmatrix}$$

13   Bandpass Predictor Filter

14           The previous section described the design of a lowpass

15   predictor. In this section, bandpass processes are considered.

16           A bandpass predictor filter is designed by modulating a

17   lowpass prototype sinc function to the required center

18   frequency $\theta_0$. The bandpass predictor coefficient $h_{BP}(n)$ is

19   obtained from the prototype lowpass sinc function $h_{LP}(n)$ as:

20   Equation 20:

21           $$\mathrm{sinc}_{BP}(n) = \mathrm{sinc}_{LP}(n)\cos(\theta_0(n-k)) \qquad n = 0,\ldots,2p$$

22

23   where $k = \left[\dfrac{2p+1}{2}\right]$.

24

25   Highpass Predictor Filter

26           A highpass predictor filter is designed by highpass

27   modulating a lowpass prototype sinc function to the required

28   corner frequency $\theta_c$. The highpass predictor coefficients $h_{HP}(n)$

1  are obtained from the prototype lowpass sinc function $h_{LP}(n)$

2  as:

3  Equation 21:

$$\text{sinc}_{HP}(n) = \text{sinc}_{LP}(n) \; (-1)^{n-k} \qquad n = 0,...,2p$$

4

5

6  ## $\Sigma\Delta$ Modulator FPGA Implementation

7         The most challenging aspect of implementing the data

8  modulator is producing an efficient implementation for the

9  prediction filter P(z). The desire to support high-sample

10  rates, and the requirement of zero latency for P(z), will

11  preclude bit-serial methods from this problem. In addition,

12  for the sake of area efficiency, parallel multipliers that

13  exploit one time-invariant input operand (the filter

14  coefficients) will be used, rather than general variable-

15  variable multipliers. The constant-coefficient multiplier

16  (KCM) is based on a multi-bit inspection version of Booth's

17  algorithm. Partitioning the input variable into 4-bit

18  nibbles is a convenient selection for the Xilinx Virtex

19  function generators (FG). Each FG has 4 inputs and can be

20  used for combinatorial logic or as application RAM/ROM. Each

21  logic slice in the Virtex logic fabric comprises 2 FGs, and

22  so can accommodate a 16 × 2 memory slice. Using the rule of

23  thumb that each bit of filter coefficient precision

24  contributes 5 dB to the sidelobe behavior, 12-bit precision

25  is used for P(z). 12-bit precision will also be employed for

26  the input samples. There are 3 4-bit nibbles in each input

27  sample. Concurrently, each nibble addresses independent 16 ×

28  16 lookup tables (LUTs). The bit growth incorporated here

29  allows for worst case filter coefficient scaling in P(z). No

1   pipeline stages are permitted in the multipliers because of

2   P(z)'s location in the feedback path of the modulator.

3        It is convenient to use the transposed FIR filter for

4   constructing the predictor. This allows the adders and delay

5   elements in the structure to occupy a single slice. 64

6   slices are required to build the accumulate-delay path. The

7   FPGA logic requirements for P(z), using a 9-tap predictor,

8   is $\Gamma(P(z)) = 9 \times 40 + 64 = 424$ CLBs. A small amount of

9   additional logic is required to complete the entire $\Sigma\Delta$

10  modulator. The final slice count is 450. The entire

11  modulator comfortably operates with a 113 MHz clock. This

12  clock frequency defines the system sample rate, so the

13  architecture can support a throughput of 113 MSamples per

14  second. The critical path through this part of the design is

15  related to the exclusion of pipelining in the multipliers.

16

17  Reduced Complexity FIR Mechanization

18       Now that the input signal is available as a reduced

19  precision sample stream, filtering can be performed using

20  area-optimized hardware. For the reasons discussed above, 4-

21  bit data samples are a convenient match for Virtex devices.

22  Figure 13 shows the structure of the reduced complexity FIR

23  filter. The coded samples $\hat{x}(n)$ are presented to the address

24  inputs of N coefficient LUTs. In accordance with the

25  modulated data stream precision, each LUT stores the 16

26  possible scaled coefficient values for one tap as shown in

27  Figure 14. An N-tap filter requires N such elements. The

28  outputs of the minimized multipliers are combined with an

29  add-delay datapath to produce the final result. The logic

30  requirement for the filter is $\Gamma(H(z)) = N\Gamma(MUL) + (N-1)\Gamma(ADD\_z^{-1})$

31  where $\Gamma(MUL)$ and $\Gamma(ADD\_z^{-1})$ are the FPGA area cost functions

1 for a KCM multiplier and an add-delay datapath component

2 respectively.

3      Using full-precision input samples without any $\Sigma\Delta M$

4 encoding, each KCM would occupy 40 slices. The total cost of

5 a direct implementation of H(z) is 7672 slices. The reduced

6 precision KCMs used to process the encoded data each consume

7 only 8 slices. Including the sigma-delta modulator the slice

8 count is 3002 for the $\Sigma\Delta$ approach. So the data re-

9 quantization approach consumes only 39% of the logic

10 resources of a direct implementation.

11

12 $\underline{\Sigma\Delta \text{ Decimators}}$

13      The procedure for re-quantizing the source data can

14 also be used effectively in an m:1 decimation filter. An

15 interesting problem is presented when high input sample

16 rates ($\geq$150 MHz) must be supported in FPGA technology. High-

17 performance multipliers are typically realized by

18 incorporating pipelining in the design. This naturally

19 introduces some latency in to the system. The location of

20 the predictor filter P(z) requires a zero-latency design.

21 (It is possible that the predictor could be modified to

22 predict samples further ahead in the time series, but this

23 potential modification will not be dealt with in the limited

24 space available.) Instead of re-quantizing, filtering and

25 decimating, which would of course require a $\Sigma\Delta$ modulator

26 running at the input sample rate, this sequence of

27 operations is re-ordered to permit several slower modulators

28 to be used in parallel. The process is performed by first

29 decimating the signal, re-quantizing and then filtering. Now

18

1   the $\Sigma\Delta$ modulators operate at the reduced output sample rate.

2   This is depicted in Figure 15. To support arbitrary center

3   frequencies, and any arbitrary, but integer, down-sampling

4   factor $m$, the bandpass decimation filter employs complex

5   weights. The filter weights are of course just the bandpass

6   modulated coefficients of a lowpass prototype filter

7   designed to support the bandwidth of the target signal.

8   Samples are collected from the A/D and alternated between

9   the two modulators. Both modulators are identical and use

10  the same predictor filter coefficients. The re-quantized

11  samples are processed by an m:1 complex polyphase filter to

12  produce the decimated signal. Several design options are

13  presented once the signal has been filtered and the sample

14  rate lowered. Figure 15 illustrates one possibility. Now

15  that the data rate has been reduced, the low rate signal is

16  easily shifted to baseband with a simple, and area

17  efficient, complex heterodyne. One multiplier and a single

18  digital frequency synthesizer could be time shared to

19  extract one or multiple channels.

20       It is interesting to investigate some of the changes

21  that are required to support the $\Sigma\Delta$ decimator. The center

22  frequency of the prediction filter should be designed to

23  predict samples in the required spectral region in

24  accordance with the output sample rate. For example,

25  consider m=2, and the required channel center frequency

26  located at 0.1 Hz, normalized with respect to the input

27  sample rate. The prediction filter should be designed with a

28  center frequency located at 0.2 Hz. In addition, the quality

29  of the prediction should be improved. With respect to the

30  output sample rate, the predictors are required to operate

1 over a wider fractional bandwidth. This implies more filter

2 coefficients in P(z). The increase in complexity of this

3 component should be balanced against the savings that result

4 in the reduced complexity filter stage to confirm that a net

5 savings in logic requirements is produced. To more clearly

6 demonstrate the approach, consider a 2:1 decimator, a

7 channel center frequency at 0.2 Hz and a 60 dB dynamic range

8 requirement.

9      Figure 16(a) shows the double-sided spectrum of the

10 input test signal. The input signal is commutated between $\Sigma\Delta_0$

11 and $\Sigma\Delta_1$ to produce the two low-precision sequences $\hat{x}_0(n)$ and

12 $\hat{x}_1(n)$. The respective spectrums of these two signals are shown

13 in Figures 16(b) and 16(c). The complex decimation filter

14 response is defined in Figure 16(d). After filtering, a

15 complex sample stream supported at the low output sample

16 rate is produced. This spectrum is shown in Figure 16(e).

17 Observe that the out-of-band components in the test signal

18 have been rejected by the specified amount and that the in-

19 band data meets the 60 dB dynamic range requirement. For

20 comparison, the signal spectrum resulting from applying the

21 processing stages in the order, re-quantize, filter and

22 decimate is shown in Figure 16(f). The interesting point to

23 note is that while the dual $\Sigma\Delta$ modulator approach satisfies

24 the system performance requirements, its out-of-band

25 performance is not quite as good as the response depicted in

26 Figure 16(f). The stopband performance of the dual modulator

27 architecture has degraded by approximately 6 dB. This can be

28 explained by noting that the shaping noise produced by each

29 modulator is essentially statistically independent. Since

30 there is no coupling between these two components prior to

1 filtering, complete phase cancellation of the modulator

2 noise cannot occur in the polyphase filter.

3

4 Discussion

5     To provide a frame of reference for the $\Sigma\Delta$ decimator,

6 consider an implementation that does not pre-process the

7 input data, but just applies it directly to a polyphase

8 decimation filter. A complex filter processing real-valued

9 data consumes double the FPGA resources of a filter with

10 real weights. For N=160, 15344 CLBs are required. This

11 Figure is based on a cost of 40 CLBs for each KCM and 8 CLBs

12 for an add-delay component.

13     Now consider the logic accounting for the dual

14 modulator approach. The area cost $\Gamma(\widehat{FIR})$ for this filter is

15

16 Equation 22:

$$\Gamma(\widehat{FIR}) = 2\Gamma(\Sigma\Delta) + \Gamma(\widehat{MUL})) + \Gamma(ACC\_z^{-1})$$

17

18 where $\Gamma\Sigma\Delta$ represents the logic requirements for one $\Sigma\Delta$

19 modulator, and $\Gamma(\widehat{MUL})$ is the logic needed for a reduced

20 precision multiplier. Using the filter specifications

21 defined earlier, and 18-tap error prediction filters, $\Gamma(\widehat{FIR})$ =

22 2 × 738 + 2 × ((160 + 159) × 8) = 6596. Comparing the area

23 requirements of the two options produces the ratio

24

25 Equation 23:

$$\lambda = \frac{\Gamma(\widehat{FIR})}{\Gamma FIR} = 6596/15344 \approx 43\%$$

26

1   So for this example, the re-quantization approach has

2   produced a realization that is significantly more area

3   efficient than a standard tapped-delay line implementation.

4

5   Center Frequency Tuning

6       For both the single-rate and multi-rate $\Sigma\Delta$ based

7   architectures, the center frequency is defined by the

8   coefficients in the predictor filter and the coefficients in

9   the primary filter. The constant coefficient multipliers can

10  be constructed using the FPGA function generators configured

11  as RAM elements. When the system center frequency is to be

12  changed, the system control hardware would update all of the

13  tables to reflect the new channel requirements. If only

14  several channel locations are anticipated, separate

15  configuration bit streams could be stored, and the FPGA(s)

16  re-configured as needed.

17

18  Bandpass $\Sigma\Delta$Ms Using Allpass Networks

19      In an earlier section we discussed how to design a

20  predicting filter for the feedback loop of a standard sigma

21  delta modulator. The predicting filter increases the order

22  of the modulator so that the modified structure has

23  additional degrees of freedom relative to a single-delay

24  noise feedback loop. These extra degrees of freedom have

25  been used in two ways, first to broaden the bandwidth of the

26  loop's noise transfer function, and second to tune its

27  center frequency. The tuning process entailed an off line

28  solution of the Normal equations which while not difficult,

29  does present a small delay and the need for a background

30  processor. We can define a sigma-delta loop with a

1　completely different architecture that offers the same

2　flexibility, namely wider bandwidth and a tunable center

3　frequency that does not require this background task. In

4　this alternate architecture, a fixed set of feedback weights

5　from a set of digital integrators defines a base-band

6　prototype filter with a desirable NTF. The filter is tuned

7　to arbitrary frequencies by attaching to each delay element

8　$z^{-1}$, a simple sub-processing element that performs a base-

9　band to band-pass transformation of the prototype filter.

10　This processing element tunes the center frequency of its

11　host prototype with a single real and selectable scalar. The

12　structure of a fourth order prototype sigma-delta loop is

13　shown in Figure 17. The time and spectrum obtained by using

14　the loop with a 4-bit quantizer is shown in Figure 18. In

15　this structure the digital integrator poles are located on

16　the unit circle at DC. The local feedback (a1 and a2)

17　separates the poles by sliding them along the unit circle,

18　and the global feedback (b1, b2, b3 and b4) places these

19　poles in the feedback path of the quantizer so they become

20　noise transfer function zeros. These zeros are positioned to

21　form an equal-ripple stop band for the NTF. The coefficients

22　selected to match the NTF pole-zero locations to an elliptic

23　high pass filter. The single sided bandwidth of this fourth

24　order loop is approximately 4% of the input sample rate.

25　　　　The low-pass to band-pass transformation for a sampled

26　data filter is achieved by substituting an all-pass transfer

27　function G(z) for the all-pass transfer function $z^{-1}$. This

28　transformation is shown in equation 24.

29

30　Equation 24:

31

$$z^{-1} \longrightarrow -z^{-1}\left(\frac{1-cz}{z-c}\right)$$

1

2    A block diagram of a digital filter with the transfer

3  function for G(z) is shown in Figure 19. Examining the left

4  hand block diagram, we find the transfer function from x(n)

5  to y(n) is the all-pass network -(1-cz)/(z-c), while the

6  transfer function from x(n) to v(n) is -(1/z)(1-cz)/(z-c).

7  When we absorb the external negative sign change in the

8  internal adders of the filter we obtain the simple right-

9  hand side version of the desired transfer function G(z).

10    After the block diagram substitution has been made, we

11  obtain Figure 20, the tunable version of the low-pass

12  prototype. The basic structure of the prototype remains the

13  same when we replace the delay with the tunable all-pass

14  network. The order of the filter is doubled by the

15  substitution since each delay is replaced by a second order

16  sub-filter. Tuning is trivially accomplished by changing the

17  c multiplier of the all-pass network. The tuned version of

18  the system reverts back to the prototype response if we set

19  c to 1.

20    Figure 21 presents the time and spectrum obtained by

21  using the tunable loop with a 4-bit quantizer shown in

22  Figure 20. The single sided bandwidth of the prototype

23  filter is distributed to the positive and negative spectral

24  bands of the tuned filter. Thus the two-sided bandwidth of

25  each spectral band is approximately 4% of the input sample

26  rate.

27    We now estimate the computational workload required to

28  operate the prototype and tunable filter. The prototype

1   filter has six coefficients to form the 4-poles and the 4-

2   zeros of the transfer function. The two $a_k$ k=0,1 coefficients

3   determine the four zero locations. These are small

4   coefficients and can be set to simple binary scalers. The

5   values computed for this filter for $a_1$ and $a_2$ were 0.0594 and

6   0.0110. These can be approximated by 1/16 and 1/128, which

7   lead to no significant shift of the spectral zeros in the

8   NTF. These simple multiplications are virtually free in the

9   FPGA hardware since they are implemented with suitable

10  wiring. The four coefficients $b_k$ k=0,…,3 are 1.000, 0.6311,

11  0.1916, and 0.0283 respectively were replaced with

12  coefficients containing one or two binary symbols to obtain

13  values 1.000, 1/2+1/8 (0.625), 1/8+1/16 (0.1875) and 1/32

14  (0.03125). When the sigma-delta loop ran with these

15  coefficients there was no discernable change in bandwidth or

16  attenuation level of the loop. The loop operates equally as

17  well in the tuning mode and the non-tuning mode with the

18  approximate coefficients listed above. Thus the only real

19  multiplies in the tunable sigma-delta loop are the c

20  coefficients of the all-pass networks. These networks are

21  unconditionally stable and always exhibit all-pass behavior

22  even in the presence of finite arithmetic and finite

23  coefficients. This is because the same coefficient forms the

24  numerator and the denominator. Errors in approximating the

25  coefficients for c simply result in a frequency shift of the

26  filter's tuned center. The c coefficient is determined from

27  the cosine of the center frequency (in radians/sample). The

28  curve for this relationship is shown in Figure 22. Also

29  shown is an error due to approximating c by $c+\delta c$. The

30  question is, what is the change in center frequency $\theta$, from

1  θ+δθ due to the approximation of c? We can see that the slope

2  at the operating point on the cosine curve is -sinθ so that

3  $\delta c/\delta\theta \approx -\sin(\theta)$ so that $\delta c \approx -\delta\theta\sin(\theta)$ is the required

4  precision to maintain a specified error. We note that tuning

5  sensitivity is most severe for small frequencies where $\sin(\theta)$

6  is near zero. The tolerance term, $\delta\theta\sin(\theta)$, is quadratic for

7  small frequencies, but the lowest frequency that can be

8  tuned by the loop is half the NTF pass-band bandwidth. For

9  the fourth order system described here, this bandwidth is 4%

10  of the sample rate, so the half-bandwidth angle is 2%, or

11  0.126 radians. To assure that the frequency to which the

12  loop is tuned has an error smaller than 1% of center

13  frequency, $\delta c < \delta\theta\sin(\theta) \Rightarrow \delta c < (0.126/100)(0.126) = 0.0002$,

14  which corresponds to a 14 bit coefficient. An error of less

15  than 10% center frequency can be achieved with 10 bit

16  coefficients.

17  The tuning} multipliers could be implemented as full

18  multipliers in the FPGA hardware or as dynamically re-

19  configured KCMs, or KDCM, as shown in Figure 23. The later

20  approach conserves FPGA resources at the expense of

21  introducing a start-up penalty each time the center

22  frequency is changed. The start-up period is the

23  initialization time of the KCM LUT. When a new center

24  frequency is desired, the tuning constant is presented to

25  the k input of the KDCM and the load signal LD is asserted.

26  This starts the initialization engine, which requires 16

27  clock cycles to initialize 16 locations in the multiplier

28  LUT. The initialization engine relies on the automatic shift

29  mode of the Virtex LUTs. In this mode of operation a LUT's

30  register contents are passed from one cell to the next cell

1   on each clock tick. This avoids the requirement for a

2   separate address generator and multiplexor in the

3   initialization hardware. Observe from Figure 23 that the

4   initialization engine only introduces a small amount of

5   additional hardware over that of a static KCM.

6        There is approximately a factor of 4 difference in the

7   area of a KDCM and full multiplier.

8

9   $\Sigma\Delta$ DC Canceler

10       Unwanted DC components can be introduced into a DSP

11  datapath at several places. It may be presented to the

12  system via an un-trimmed offset in the analog-to-digital

13  conversion pre-processing circuit, or may be attributed to

14  bias in the A/D converter itself. Even if the sampled input

15  signal has a zero mean, DC content can be introduced though

16  arithmetic truncation processes in the fixed-point datapath.

17  For example, in a multi-stage multi-rate filter, the

18  intermediate filter output samples may be quantized between

19  stages in order to compensate for the filter processing gain

20  and thereby keep the word-length requirements manageable.

21  The introduced DC bias can impact the dynamic range

22  performance of a system and potentially increase the error

23  rate in a digital receiver application.

24       In a fixed-point datapath, the bias can cause

25  unnecessary saturation events that would not occur if the DC

26  was not present in the system. In a digital communication

27  receiver employing M-ary QAM modulation, the DC bias can

28  interfere with the symbol decision process, so causing

29  incorrect decoding and therefore increasing the bit error

30  rate.

1    In some cases the introduced bias can be ignored and is

2    of no concern. However, for other applications it is

3    desirable to remove the DC component.

4        One solution to removing the unwanted DC level is to

5    employ a DC canceler. A simple canceler is shown in Figure

6    24. It is easy to show that the transfer function of the

7    network is

8

9    Figure 25:

$$H(z) = \frac{z-1}{z-(1-z)}$$

10

11

12    The cancellation is due to the transfer function zero at 0

13    Hz. The pole at 1-$\mu$ controls the system bandwidth, and hence

14    the system transient response. The location of the zero at

15    z=1 removes the DC

16    component in the signal, but there are some problems with a

17    practical implementation of this circuit.

18        Figure 25A is a spectral domain representation of a

19    biased signal presented to the DC canceler. Figure 25B is

20    the processed signal spectrum at $y_q(n)$ in Figure 24. We

21    observe that the DC content in the input signal has been

22    completely removed. However, in the process of running the

23    canceling loop the network processing gain has caused a

24    dynamic range expansion. So although the sample stream $y_q(n)$

25    is a zero mean process, it requires a larger number of bits

26    to represent each sample than is desirable. The only option

27    with the circuit is to re-quantize $y_q(n)$ to produce y(n)

28    using the quantizer Q($\cdot$). The effect of this operation is

1   shown in Figure 25C, which demonstrates, not surprisingly,

2   that after an 8-bit quantizer, the signal now has a DC

3   component and we are almost back to where we started. How

4   can the canceler be re-organized to avoid this

5   implementation pitfall? One option is to embed the re-

6   quantizer in the feedback loop in the form of a $\Sigma\Delta$ modulator

7   as shown in Figure 26. The modulator can be a very simple

8   1st order loop such as the error feedback $\Sigma\Delta$ modulator shown

9   in Figure 9. Figure 25D demonstrates the operation of the

10  circuit for 8-bit output data. Observe from the Figure that

11  the DC has been removed from the signal while employing the

12  same 8-bit output sample precision that was used in Figure

13  24. The simple $\Sigma\Delta M$

14  employed in the canceler is easily implemented in an FPGA.

15

16  Simplify Digital Receiver Control Loops Using $\Sigma\Delta$ Modulators

17        In earlier sections we recognized that when a sampled

18  data input signal has a bandwidth that is a small fraction

19  of its sample rate the sample components from this

20  restricted bandwidth are highly correlated. We took

21  advantage of that correlation to use a digital sigma-delta

22  modulator to requantize the signal to a reduced number of

23  bits. The sigma-delta modulator encodes the input signal

24  with a reduced number of bits while preserving full input

25  precision over the signal bandwidth by placing the increased

26  noise due to requantization in out-of-band spectral

27  positions that are already scheduled to be rejected by

28  subsequent DSP processing. The purpose of this

29  requantization is to allow the subsequent DSP processing to

30  be performed with reduced arithmetic resource requirements

1   since the desired data is now represented by a smaller

2   number of bits.

3       A similar remodulation of data samples can by be

4   employed for signals generated within a DSP process when the

5   bandwidth of the signals are small compared to the sample

6   rate of the process. A common example of this circumstance

7   is the generation of control signals used in feedback paths

8   of a digital receiver. These control signals include a gain

9   control signal for a voltage controlled amplifier in an

10  automatic gain control (AGC) loop and VCO (voltage

11  controlled oscillator) control signals in carrier recovery

12  and timing recovery loops. A block diagram of a receiver

13  with these specific controls signals is shown in Figure 27.

14  The control signals are generated from processes operating

15  at a sample rate appropriate to the input signal bandwidth.

16  The bandwidth of control loops in a receiver are usually a

17  very small fraction of the signal bandwidth, which means

18  that the control signal are very heavily oversampled. As a

19  typical example, in a cable TV modem, the input bandwidth is

20  6 MHz, the processing sample rate is 20 MHz, and the loop

21  bandwidth may be 50 kHz. For this example, the ratio of

22  sample rate to bandwidth is 400-to-1.

23      As seen in Figure 27, the process of delivering these

24  oversampled control signals to their respective control

25  points entails the transfer of 16 bit words to external

26  control registers, requiring appropriate busses, addressing,

27  and enable lines as well as the operation of 16-bit digital-

28  to-analog converters (DACs).

29      We can use a sigma-delta modulator to requantize the

30  16-bit oversampled control signals in the digital receiver

31  prior to passing them out of the processing chip. The sigma-

1   delta can preserve the required dynamic range over the

2   signal's restricted bandwidth with a one-bit output. As

3   suggested in Figure 28, the transfer of a single bit to

4   control the analog components is a significantly less

5   difficult task than the original. We no longer require

6   registers to accept the transfer, the busses to deliver the

7   bits, or the DAC to convert the digital data to the analog

8   levels the data represents. All that is needed a simple

9   filter (and likely an analog amplifier to satisfy drive

10  level and offset requirements). Experience shows that a 1-

11  bit, one-loop sigma-delta modulator could achieve 80 dB

12  dynamic range and requires a single RC filter to reconstruct

13  the analog signal. A two-loop sigma-delta modulator is

14  required to achieve 16-bit precision for which a double RC

15  filter is required to reconstruct the analog output signal.

16  Figure 29 shows the time response of the one-bit two loop

17  sigma-delta converter to a slowly varying control signal and

18  the reconstructed signal obtained from the dual-RC filter.

19  Figure 30 shows the spectrum obtained from a 1-bit two-loop

20  modulator and the spectrum obtained from an unbuffered RC-RC

21  filter.

22      This example has shown how with minimal additional

23  hardware, an FPGA can generate analog control signals to

24  control low-bandwidth analog functions in a system.

25  An observation worthy of note, is that the audio engineering

26  community has recognized the advantage offered by this

27  option of requantizing a 16-bit oversampled data stream to

28  1-bit data stream. In that community, the output signal is

29  intentionally upsampled by a factor of 64 and then

30  requantized to 1-bit in a process called a MASH converter.

1   Nearly all CD players use the MASH converter to deliver

2   analog audio signals.

3

4   <u>What Have We Gained?</u>

5        What has been achieved by expressing our signal

6   processing problems in terms of $\Sigma\Delta$M techniques?

7

8        The paper has demonstrated some $\Sigma\Delta$M techniques for the

9   compact implementation of certain types of filter and

10  control applications using FPGAs. This optimization can be

11  used in several ways to bring economic benefits to a

12  commercial design. By exploiting $\Sigma\Delta$M filter processes, a

13  given processing load may be realizable in a lower-density,

14  and hence less expensive, FPGA than is possible without

15  access to these techniques. An alternative would be to

16  perform more processing using the same hardware. For

17  example, processing multiple channels in a communication

18  system.

19       In addition to FPGA area trade-offs, the $\Sigma\Delta$M methods

20  can result in reduced power consumption in a design. Power P

21  may be expressed as

22

$$P = CV^2 f_{clk}$$

23  where C is capacitance, V is voltage and f is the system

24  clock frequency. By reducing the silicon area requirements

25  of a filter, we can simultaneously reduce the power

26  consumption of the design. For the examples considered

27  earlier, logic resource savings of greater than 50% were

28  demonstrated. The savings is proportional to increased

32

1 efficiency in the system power budget, and this of course is

2 very important for mobile applications.

3     The $\Sigma\Delta M$ AGC, timing and carrier recovery control loop

4 designs are also important examples in a industrial context.

5 The examples illustrated how the component count in a mixed

6 analog/digital system can be reduced. In fact, not only is

7 the component count reduced, but printed circuit board area

8 is minimized. This results in more reliable and physically

9 smaller implementations. The reduced component count also

10 results in reduced power consumption. In addition, since the

11 control loops no longer require wide output buses from the

12 FPGA to multi-bit DACs that generate analog control

13 voltages, power consumption is decreased because fewer FPGA

14 I/O pads are being driven.

15

16 <u>References</u>

17     The subject matter of this application is excerpted

18 from and article entitled "FPGA Signal Processing Using

19 Sigma-Delta Modulation," C. H. Dick and F. J. Harris, IEEE

20 Signal Processing Magazine (January 2000), which is

21 incorporated herein by reference. The following documents

22 may also be of interest, and are also incorporated by

23 reference.

24 [1]    Atmel, AT40K/05/10/20/40 Data Sheet, 1999.

25 [2]    J. A. C. Bingham, The Theory and Practice of Modem

26        Design, John Wiley & Sons, New York, 1998.

27 [3]    J. C. Candy and G. C. Temes, "Oversampling Methods for

28        A/D and D/A Conversion" in Oversampling Delta Sigma

29        Data Converters - Theory Design and Simulation, IEEE

30        Press, New York, 1992.

1  [4]  M. E. Frerking, Digital Signal Processing in
2       Communication Systems, Van Nostrand Reinhold, New York,
3       1994.
4  [5]  S. Haykin, "Modern Filters", Macmillan Publishing Co.,
5       New York, 1990.
6  [6]  H. Meyr, M. Moeneclaey and S. A. Fechtal, Digital
7       Communication Receivers, John Wiley & Sons Inc., New
8       York, 1998.
9  [7]  S. R. Norsworthy, R. Schreier and G. C. Temes, \Delta-
10      Sigma Data Converters", IEEE Press, Piscataway, NJ,
11      1997.
12 [8]  D. A. Patterson and J. L. Hennessy, Computer
13      Architecture: A Quantitative Approach, Morgan Kaufmann
14      Publishers Inc., California, 1990.
15 [9]  A. Peled and B. Liu, "A New Hardware Realization of
16      Digital Filters", IEEE Trans. on Acoust., Speech,
17      Signal Processing, vol. 22, pp. 456-462, Dec. 1974.
18 [10] J. G. Proakis, and D. G. Manolakis, Digital Signal
19      Processing Principles, Algorithms and Applications
20      Second Edition, Maxwell Macmillan International, New
21      York, 1992.
22 [11] S. A. White, "Applications of Distributed Arithmetic to
23      Digital Signal Processing", IEEE ASSP Magazine, Vol.
24      6(3), pp. 4-19, July 1989.
25 [12] Xilinx Inc., The Programmable Logic Data Book, 1999.